Fundamenta Informaticae XX (2008) 1–15
IOS Press

Constraint Based World Modeling

Daniel Göhring, Heinrich Mellmann, Kataryna Gerasymova, and Hans-Dieter Burkhard
Institut für Informatik
Humboldt-Universität
Unter den Linden 6, 10099 , Berlin, Germany
{*goehring / mellmann / gerasymo / hdb*}*@informatik.hu-berlin.de*

Abstract. Common approaches for robot navigation use Bayesian filters like particle filters, Kalman filters and their extended forms. We present an alternative and supplementing approach using constraint techniques based on spatial constraints between object positions. This yields several advantages. The robot can choose from a variety of belief functions, and the computational complexity is decreased by efficient algorithms. The paper investigates constraint propagation techniques under the special requirements of navigation tasks. Sensor data are noisy, but a lot of redundancies can be exploited to improve the quality of the result. We introduce two quality measures: The ambiguity measure for constraint sets defines the precision, while inconsistencies are measured by the inconsistency measure. The measures can be used for evaluating the available data and for computing best fitting hypothesis. A constraint propagation algorithm is presented.

Keywords: Robot Navigation, Constraints

1. Introduction

Modeling the world state is important for many robot tasks. Standard methods for world modeling use Bayesian approaches [7] like Kalman filters [9] or particle filters. The drawbacks of Kalman filters are the underlying assumptions about the statistics (approximations of Gaussian distributions). Particle filters are limited because of time complexity. For that, they usually do not use all available data, but only focus on particular information. Moreover, they cannot make best hypothesis guesses.

Given an image of a scene, we have constraints between the objects in the image and the objects in the scene. Object parameters, image parameters and camera parameters are dependent by related constraints. They can easily be combined with constraints derived from other sensor measurements [8].

Address for correspondence: Daniel Göhring, LFG Künstliche Intelligenz, Institut für Informatik, HU-Berlin, Unter den Linden 6, 10099 Berlin, Germany

Given odometry (or control) data, subsequent positions are constrained by measured speed and direction of movements.

Constraints can have several advantages: Their definition is formal and they are easy to communicate - which is in contrast to other concepts representing beliefs, e.g., particle distributions. Constraint based modeling approaches have been proposed for localization in [11], or for slam map building in [6], [10].

In this paper we discuss constraint propagation methods for solving navigation problems. The main difference to classical propagation is due to the fact that navigation tasks do always have a solution in reality. For that, inconsistencies have to be resolved e.g. by relaxing constraints. Moreover, navigation tasks are not looking for a single solution of the constraint problem. Instead, all possible solutions are interesting in order to know about the ambiguity of the solution (which is only incompletely addressed by particle filters). For that, the notion of conservative propagation functions is introduced. It can be shown that this notion coincides to some extend to the classical notion of local consistency (for maximal locally consistent intervals).

We have to deal with incomplete or with noisy measurements. With incomplete measurements, the result of constraint propagation will be ambiguous, while noisy measurements may lead to inconsistent constraints. We therefore introduce measures of ambiguity and inconsistency and investigate the usefulness of constraints: Some constraints which are not useful or which are in conflict with others might be discarded.

The paper is structured as follows: Section 2 gives an introduction and an example for description and usage of constraints generated from sensor data. In Section 3 we present the formal definitions and the backgrounds for usage of constraints. The quality measures are introduced and discussed in Sections 4 and 5. Basics of constraint propagation in the context of navigation tasks are discussed in Section 6, and an efficient algorithm is presented.

For illustration purposes we will consider navigation problems of soccer playing robots from the RoboCup competitions [2], where we have investigated several aspects of perception and control over the last years (for a collection of related papers see [1]).

2. Perceptual Constraints

A constraint C is defined over a set of variables $v(1), v(2), ..., v(k)$. It defines the values those variables can take:

$$C \subseteq Dom(v(1)) \times ... \times Dom(v(k))$$

We start with an example from RoboCup where the camera image of a robot shows a goal in front and the ball before the white line of the penalty area (Figure 1). It is not too difficult for a human interpreter to give an estimate for the position (x_B, y_B) of the ball and the position (x_R, y_R) of the observing robot. Humans can do that, regarding relations between objects, like the estimated distance d_{BR} between the robot and the ball, and by their knowledge about the world, like the positions of the goalposts and of the penalty line.

The program of the robot can use the related features using image processing. The distance d_{BR} can be calculated from the size of the ball in the image, or from the angle of the camera. The distance d_{BL} between the ball and the penalty line can be calculated, too. Other values are known parameters of the environment: $(x_{Gl}, y_{Gl}), (x_{Gr}, y_{Gr})$ are the coordinates of the goalposts, and the penalty line is given as the set of points $\{(x, b_{PL}) | -a_{PL} \leq x \leq a_{PL}\}$. The coordinate system has its origins at the center point, the y-axis points to the observed goal.

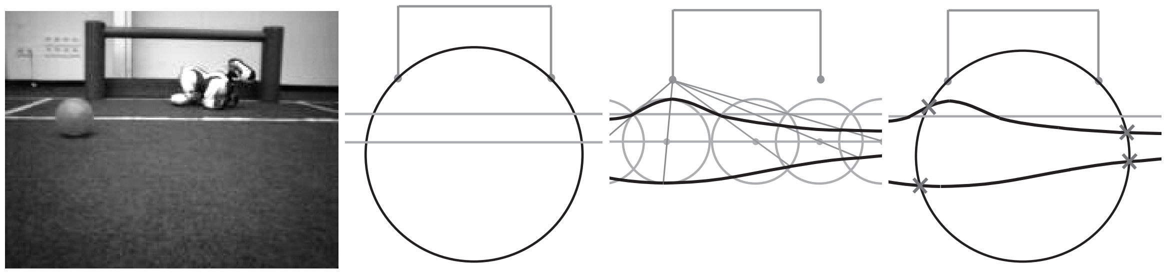

Figure 1. Example from RoboCup (Four legged league): A robot is seeing a goal and the ball before the penalty line. The first Figure on left side illustrates the scene from the view of the robot. The next three figures show the related constraints that can be used for localization. **Left**: The picture shows a part of the field with the goal and the white penalty line, the periphery circle according to C_1, and the line of the Ball-Line-Constraint C_2. **Middle**: The picture shows the Constraint C_2 for the ball, some of the circles according to constraint C_5, some of the lines according to C_4, and the resulting two lines for C_6. **Right**: Constraints according to C_7: The position of the robot is one of the four intersection points between the periphery circle (C_1) and the lines according to C_6.

The relations between the objects can be described by constraints. The following four constraints are obvious by looking to the image, and they can be determined by the program of the observing robot:

C_1**:** The view angle γ between the goalposts (the distance between them in the image) defines a circle (periphery circle), which contains the goal posts coordinates $(x_{Gl}, y_{Gl}), (x_{Gr}, y_{Gr})$ and the coordinates (x_R, y_R) of the robot:

$$\{(x_R, y_R) | \arctan \frac{y_{Gl} - y_R}{x_{Gl} - x_R} - \arctan \frac{y_{Gr} - y_R}{x_{Gr} - x_R} = \gamma\}$$

C_2**:** The ball lies in the distance d_{BL} before the penalty line. Therefore, the ball position must be from the set

$$\{(x_B, y_B) | x_B \in [-a_{PL}, a_{PL}], y_B = b_{PL} - d_{BL}\}$$

C_3**:** The distance d_{BR} between the robot and the ball defines a circle such that the robot is on that circle around the ball:

$$(x_B - x_R)^2 + (y_B - y_R)^2 = d_{BR}^2$$

.

C_4**:** The observer, the ball and the left goal post are on a line:

$$\frac{x_R - x_B}{y_R - y_B} = \frac{x_B - x_{Gl}}{y_B - y_{Gl}}$$

The points satisfying the constraints by C_1 (for the robot) and by C_2 (for the ball) can be visualized immediately on the playground as in Figure 1 (left).

The constraint by C_3 does not give any restriction to the position of the ball. The ball may be at any position on the playground, and then the robot has a position somewhere on the circle around the ball. Or vice versa for reasons of symmetry: The robot is on any position of the playground, and the ball around him on a circle. In fact, we have four variables which are restricted by C_3 to a subset of a four dimensional space. The same applies to constraint C_4.

The solution (i.e. the positions) must satisfy all four constraints. We can consider all constraints in the four dimensional space of the variables (x_B, y_B, x_R, y_R) such that each constraint defines a subset of this space. Then we get the following constraints:

$$C_1 = \{(x_B, y_B, x_R, y_R) | \arctan \frac{y_{Gl} - y_R}{x_{Gl} - x_R} - \arctan \frac{y_{Gr} - y_R}{x_{Gr} - x_R} = \gamma\} \tag{1}$$

$$C_2 = \{(x_B, y_B, x_R, y_R) | (x_B \in [-a_{PL}, a_{PL}], y_B = b_{PL} - d_{BL}\} \tag{2}$$

$$C_3 = \{(x_B, y_B, x_R, y_R) | (x_B - x_R)^2 + (y_B - y_R)^2 = d_{BR}^2\} \tag{3}$$

$$C_4 = \{(x_B, y_B, x_R, y_R) | \frac{x_R - x_B}{y_R - y_B} = \frac{x_B - x_{Gl}}{y_B - y_{Gl}}\} \tag{4}$$

Then the possible solutions (as far as determined by C_1 to C_4) are given by the intersection $\bigcap_{1,\ldots,4} C_i$. According to this fact, we can consider more constraints $C_5, \ldots, C_n$ as far as they do not change this intersection, i.e. as far as $\bigcap_{1,\ldots,n} C_i = \bigcap_{1,\ldots,4} C_i$. Especially, we can combine some of the given constraints.

By combining C_2 and C_3 we get the constraint $C_5 = C_2 \cap C_3$ where the ball position is restricted to any position on the penalty line, and the player is located on a circle around the ball. Then, by combining C_4 and C_5 we get the constraint $C_6 = C_4 \cap C_5$ which restricts the positions of the robot to the two lines shown in Figure 1 (middle).

Now intersecting C_1 and C_6 we get the constraint C_7 with four intersection points as shown in Figure 1 (right). According to the original constraints C_1 to C_4, these four points are determined as possible positions of the robot. The corresponding ball positions are then given by C_2 and C_4.

To find the real positions, we would need additional constraints from the image, e.g. that the ball lies between the robot and the goal (which removes one of the lines of C_6, cf. Figure 2), and that the robot is located on the left site of the field (by exploiting perspective).

3. Formal Definitions of Constraints

We define all constraints over the set of all variables $v(1), v(2), ..., v(k)$ (even if some of the variables are not affected by a constraint). The domain of a variable v is denoted by $Dom(v)$, and the whole universe under consideration is given by

$$U = Dom(v(1)) \times \cdots \times Dom(v(k))$$

For this paper, we will consider all domains $Dom(v)$ as (may be infinite) intervals of real numbers, i.e. $U \subseteq \mathbb{R}^k$.

Definition 3.1. (Constraints)

1. A **constraint** C over $v(1), ..., v(k)$ is a subset $C \subseteq U$.

2. An assignment β of values to the variables $v(1), ..., v(k)$, i.e. $\beta \in U$, is a **solution** of C iff $\beta \in C$.

Definition 3.2. (Constraint Sets)

1. A **constraint set** $\mathcal{C}$ over $v(1), ..., v(k)$ is a finite set of constraints over those variables: $\mathcal{C} = \{C_1, ..., C_n\}$.

2. An assignment $\beta \in U$ is a **solution** of $\mathcal{C}$ if β is a solution of all $C \in \mathcal{C}$, i.e. if $\beta \in \bigcap \mathcal{C}$.

3. A constraint set $\mathcal{C}$ is **inconsistent** if there is no solution, i.e. if $\bigcap \mathcal{C}$ is empty.

The problem of finding solutions is usually denoted as solving a constraint satisfaction problem (CSP) which is given by a constraint set $\mathcal{C}$. By our definition, a solution is a point of the universe U, i.e. an assignment of values to all variables. For navigation problems it might be possible that only some variables are of interest. This would be the case if we are interested only in the position of the robot in our example above. Nevertheless we had to solve the whole problem to find a solution.

In case of robot navigation, there is always a unique solution of the problem in reality (the positions in the real scene). This has an impact on the interpretation of solutions and inconsistencies of the constraint system (cf. Section 6.1).

The constraints are models of relations (restrictions) between objects in the scene. The information can be derived from sensory data, from communication with other robots, and from knowledge about the world – as in the example from above. Since information may be noisy, the constraints may not be as strict as in the introductory example from Section 2. Instead of a circle we get an annulus for the positions of the robot around the ball according to C_3 in the example. In general, a constraint may concern a subspace of any dimension (e.g. the whole penalty area, the possible positions of an occluded object, etc.). Moreover, constraints need not to be connected: If there are indistinguishable landmarks, then the distance to such landmarks defines a constraint consisting of several circles. Further constraints are given by velocities: changes of locations are restricted by the direction and speed of objects.

There are many redundancies which are due to all available constraints. Visual information in images usually contains lots of useful information: Size and appearance of observed objects, bearing angles, distances and other relations between observed objects, etc. Only a very small part of this information is usually used in classical localization algorithms. This might have originated in the fact, that these algorithms have been developed for range measurements. Another problem is the large amount of necessary calculation for Bayesian methods (grids, particles). Kalman filters can process such large amounts, but they rely on additional presumptions according to the underlying statistics.

Like Kalman filters, the constraint approach has the advantage, that it can simultaneously compute positions of different objects and the relations between them. Particle filters can deal only with small dimensions of search spaces.

For constraint methods, we have special problems with inconsistencies. According to the noise of measurements, it may be impossible to find a position which is consistent with all constraints. In our formalism. the intersection of all constraints will be empty in such a case. Inconsistency in constraint satisfaction problems means usually that there does not exist a solution in reality. But in our situation, the robot (and the other objects) do have their coordinates, only the sensor noise corrupted the data. Related quality measures for constraint sets have been presented in [4]. They will be investigated in the following section..

4. Quality Measures for Constraint Sets

If all sensory measurements are precise, we can hope to get a well defined solution for the navigation problems, i.e. exact positions of the interesting objects. Furthermore we must have enough constraints to restrict the possible solutions to only a single solution. Otherwise we had to deal with ambiguities. Hence there are two aspects for the quality of a constraint set: Its consistency and its ambiguity. Both are subject to trade offs (cf. Section 5). We will define measures for inconsistency and ambiguity in the following.

4.1. Inconsistency Measure

A constraint system $\mathcal{C} = \{C_1, C_2, ..., C_n\}$ is inconsistent if there is no solution according to Definition 3.2, i.e. if $\bigcap \mathcal{C}$ is empty. Now a measure for inconsistency should reflect "how far" $\mathcal{C}$ is from having a solution.

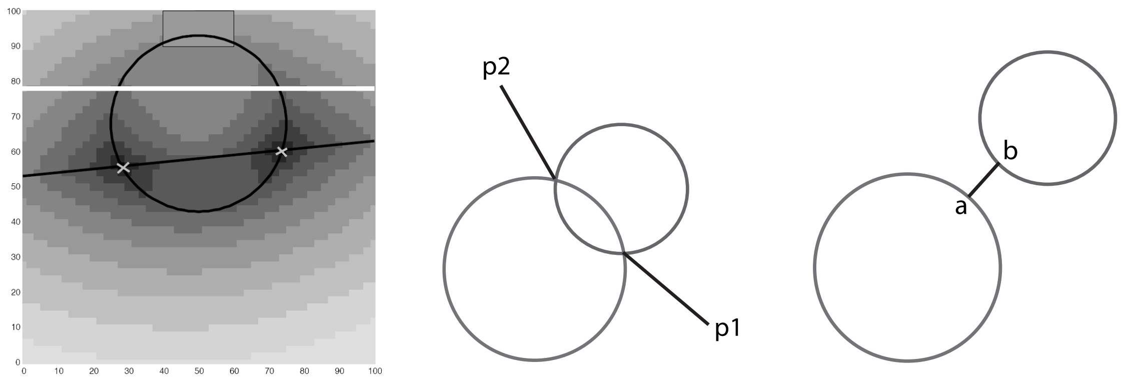

Figure 2. Left: The points of the same lightness have the same sum of distances to all constraints. The smaller s is, the darker are the points. The sets $P_{\mathcal{C}}(s)$ are built from points with lightness corresponding to s or to a value smaller than s (darker points). The picture corresponds to the right part of Figure 1 with only one of the lines for C_6 (the one for the correct order of robot, player and goal post). Middle and Right: Examples for IK. Left: $IK(\mathcal{C}) = 0$. Right: $IK(\mathcal{C}) > 0$

Definition 4.1. (Distances)
Let $d(p, p')$ be a distance between points p and p' in the universe U (we will use the Euclidean distance in our examples).

1. The distance between any point $p \in U$ and a constraint $C \subseteq U$ is given by

$$d(p, C) = \underset{p_c \in C}{min}\{d(p, p_c)\}, \tag{5}$$

2. The distance between any point $p \in U$ and a constraint set $\mathcal{C} = \{C_1, \ldots, C_n\}$ is given by

$$d(p, \mathcal{C}) = \sum_{i=1}^{n} d(p, C_i) \tag{6}$$

3. The set of all points $p \in U$ where the distance to the constraint set $\mathcal{C}$ is equal or less s is defined by
$$P_{\mathcal{C}}(s) = \{p | d(p, \mathcal{C}) \leq s\} \tag{7}$$

There are visualizations of sets $P_{\mathcal{C}}(s)$ on Figure 2 (left). As a corollary we find

Corollary 4.1. A constraint set $\mathcal{C}$ is consistent iff $P_{\mathcal{C}}(0)$ is not empty.

This leads to the following definition of the inconsistency measure for constraint sets $\mathcal{C}$:

Definition 4.2. (Inconsistency Measure)
$$IK(\mathcal{C}) = min(\{s | P_{\mathcal{C}}(s) \neq \emptyset\}) \tag{8}$$

The geometrical interpretation of the inconsistency measure is the shortest distance the constraints of the system $\mathcal{C}$ have to be moved to in order this system to become consistent:

Corollary 4.2.
$$IK(\mathcal{C}) = \underset{p}{min} \sum_{i=1}^{n} d(p, C_i) \tag{9}$$

Examples with two circular constraints are given in Figure 2. If both circles have intersecting points (middle), then the system is consistent with $IK = 0$. But there might be some ambiguity. We will discuss such situations below in the following subsection. The right example shows an inconsistent system with $IK > 0$. All points laying on the line between points a and b have the same (Euclidean) distance to the constraint set. We have $IK(\mathcal{C}) = d(a, b)$ and $P_{\mathcal{C}}(IK(\mathcal{C}))$ is the line between a and b.

In case of $IK(C) > 0$ we have no solution, but for small values of $IK(\mathcal{C})$ we could use $P_{\mathcal{C}}(IK(\mathcal{C}))$ instead. For larger inconsistencies, we could try to find some consistent subsets of the constraint set. But it might then happen, that the solution becomes more ambiguous. In the following we propose an ambiguity measure for constraint sets.

4.2. Ambiguity Measures

Without constraints, any $p \in U$ is a possible solution, and having only a single constraint C, all $p \in C$ are candidates for the solution of the navigation problem. Differently to other constraint satisfaction problems, the robot has a certain position in reality – this is what we are looking for. If the constraints do not allow an exact determination of the position, we are left with some ambiguity. Actually, the robot needs to solve its navigation problem in order to fulfil some tasks. Therefore some ambiguity may be without consequences. It might be enough to know that the robot is in a certain area (e.g. for avoiding offside in soccer).

On the other hand, having an inconsistent constraint set $\mathcal{C}$, we have no candidates for a solution. But since we know that the robot is somewhere in the environment, we can look for a nonempty set $P_{\mathcal{C}}(s)$. Those sets with minimal parameter s would be the best guesses for the navigation problem, especially $P_{\mathcal{C}}(IK(\mathcal{C}))$.

We will hence give a definition of an ambiguity measure for any subset $P \subseteq U$. There are different aspects to be considered. The volume of P could be a measure for the total amount of possible solutions.

If P is disconnected, then the number of connected components, or the distance between the components provide other measures. As discussed above, it depends on the task of the robot and on the situation which measure is adequate. For the studies in this paper, we will consider only one of these measures as an example. Actually, the only property which is important for our results is monotonicity: The ambiguity increases if the set P becomes larger.

Definition 4.3. (Ambiguity)
For a nonempty set $P \subseteq U$ we define the ambiguity by

$$Amb(P) = max\{d(p, p')|p, p' \in P\} \tag{10}$$

For technical reasons, we define $Amb(\emptyset) = -1$ for the empty set.

By this definition, the ambiguity is zero if and only if P contains a single point.

5. Optimal Constraint Sets

Using more constraints can reduce ambiguity but increases inconsistency: We have a trade-off which affects the use of available constraints. In fact, a lot of different sensory data and other information can be used for navigation purposes, but they may contradict each other to some extend by sensory noise or processing errors. It may be useful to accept a certain degree of inconsistency for having less ambiguity. As discussed in Section 4.2, it depends on the tasks of the robots which kind of ambiguity is acceptable.

Therefore it is necessary to evaluate constraint sets $\mathcal{C}$ in order to decide for using more or less constraints. We will consider the consequences of changing constraint sets $\mathcal{C}$ for inconsistency and ambiguity in the following.

Obviously, the set of solutions decreases for more constraints:

$$\mathcal{C} \subseteq \mathcal{C}' \Rightarrow \bigcap \mathcal{C} \supseteq \bigcap \mathcal{C}' \tag{11}$$

More generally, we have for any number s:

$$\mathcal{C} \subseteq \mathcal{C}' \Rightarrow P_{\mathcal{C}}(s) \supseteq P_{\mathcal{C}'}(s) \tag{12}$$

Concerning our quality measures we have

Proposition 5.1.

$$\mathcal{C} \subseteq \mathcal{C}' \Rightarrow IK(\mathcal{C}) \leq IK(\mathcal{C}') \tag{13}$$

$$\mathcal{C} \subseteq \mathcal{C}' \Rightarrow Amb(\bigcap \mathcal{C}) \geq Amb(\bigcap \mathcal{C}') \tag{14}$$

The proposition shows a classical trade-off between inconsistency and ambiguity for the choice of constraints to be used: The larger we choose the constraint set $\mathcal{C}$ from a set of available constraints, the more decreases the ambiguity of the solution set and the more increases the inconsistency of the system. Actually the solution set becomes empty (and ambiguity becomes simply -1) if inconsistency reaches a value greater than zero (if the system becomes inconsistent). Hence, to consider only this trade-off is not really helpful for our purposes.

For our purposes, the sets $P_{\mathcal{C}}(IK(\mathcal{C}))$ are more important since we can use them as substitutes for the solution sets $\bigcap \mathcal{C}$ in case of inconsistent constraint sets $\mathcal{C}$. At first sight, it might seem that the sets $P_{\mathcal{C}}(IK(\mathcal{C}))$ decrease as well if the constraint sets $\mathcal{C}$ become larger in a similar way as for the solutions in (11) from above. Then the ambiguity of these sets would decrease as well similarly to (14). But the sets $P_{\mathcal{C}}(IK(\mathcal{C}))$ do not necessarily decrease if the constraint sets become larger:

Proposition 5.2. There exist constraint sets $\mathcal{C} \subseteq \mathcal{C}'$ such that
$P_{\mathcal{C}}(IK(\mathcal{C})) \subseteq P_{\mathcal{C}}(IK(\mathcal{C}'))$ and hence $Amb(P_{\mathcal{C}}(IK(\mathcal{C}))) \leq Amb(P_{\mathcal{C}}(IK(\mathcal{C}')))$.

As an example we consider constraint sets $\mathcal{C} = \{\{p\}\}$ and $\mathcal{C}' = \{\{p\}, \{p'\}\}$ for different points $p, p' \in U$. Then we have $IK(\mathcal{C}) = 0$ and $IK(\mathcal{C}') = d(p, p')$, respectively. Therefore we get $P_{\mathcal{C}}(IK(\mathcal{C})) = \{p\} \subseteq \{p, p'\} \subseteq P_{\mathcal{C}'}(IK(\mathcal{C}'))$ which shows the proposition.

To go more into the details: We have indeed $P_{\mathcal{C}}(s) \supseteq P_{\mathcal{C}'}(s)$ for $\mathcal{C} \subseteq \mathcal{C}'$ by (12), but now the values of s may increase if we use $s = IK(\mathcal{C})$ for inconsistent constraint sets. For such cases we have

Proposition 5.3.

$$s \leq s' \Rightarrow P_{\mathcal{C}}(s) \subseteq P_{\mathcal{C}}(s') \tag{15}$$

$$s \leq s' \Rightarrow Amb(P_{\mathcal{C}}(s)) \leq Amb(P_{\mathcal{C}}(s')) \tag{16}$$

Figure 3 shows on the right the values of $IK(\mathcal{C})$ and $Amb(P_{\mathcal{C}}(IK(\mathcal{C})))$ for all nonempty subsets $\mathcal{C}$ of the constraint set $\mathcal{C}' = \{C_1, \ldots, C_4\}$ from the left. The values for the sequence $\{C_1\} \subset \{C_1, C_2\} \subset \{C_1, C_2, C_3\} \subset \{C_1, C_2, C_3, C_4\}$ do not lie on a monotonous line in the diagram: Using more constraints may increase the ambiguity values $Amb(P_{\mathcal{C}}(IK(\mathcal{C})))$. The example illustrates the fact that a more detailed analysis is necessary to find an optimal set of constraints.

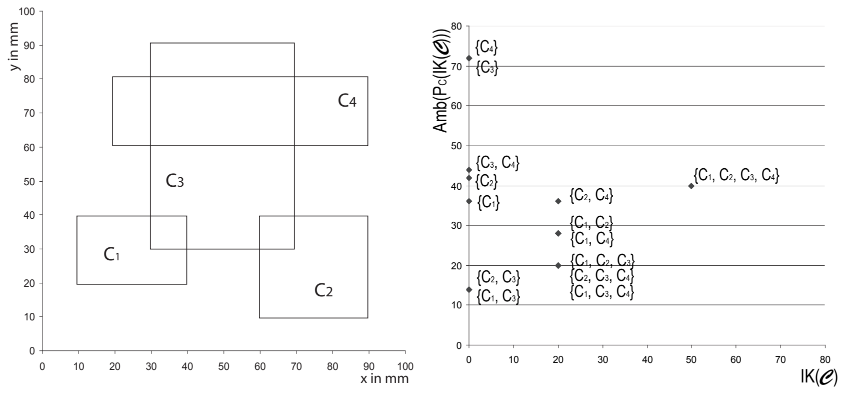

Figure 3. **Left**: Example Constraint set $\mathcal{C}$. **Right**: Inconsistency and Ambiguity measure for elements of the potential set of $\mathcal{C}$

At the end of this section we have to remark that the results for ambiguity use only the monotony property: If $P \subseteq P'$, then $Amb(P) \leq Amb(P')$. Therefore the results would be true for other kinds of ambiguity measures, too.

6. Algorithms

To use constraints for navigation problems, we need related algorithms for computing the solutions of a constraint problem or to calculate $P_{\mathcal{C}}(IK(\mathcal{C}))$ as the "best" fitting set of hypothesis for the positions of objects in the case of inconsistencies. Alternatively, to find useful constraints we can compare different sets of constraints by their values concerning $IK(\mathcal{C})$ and $Amb(P_{\mathcal{C}}(IK(\mathcal{C})))$.

In principle, many of the problems can be solved by grid based techniques. For each grid cell we can test if constraints are satisfied. This corresponds to some of the known Bayesian techniques including particle filters. The quality measures can be calculated as well for grid cells, since the basic operations are distances, sums, and minima/maxima, respectively. But such techniques are far to complex for higher dimensions of the space of the variables $v(1), \ldots, v(n)$. Gradient descent methods are an alternative (with problems of local minima).

Another alternative are techniques from constraint propagation. We can successively restrict the domains of variables by combining constraints. We will discuss constraint propagation in the following subsection.

6.1. Constraint Propagation

Known techniques (cf. e.g. [3] [5]) for constraint problems produce successively reduced sets leading to a sequence of decreasing restrictions

$$U = D_0 \supseteq D_1 \supseteq D_2, \supseteq \ldots$$

Restrictions for numerical constraints are often considered in the form of k-dimensional intervals $I = [a,b] := \{x | a \leq x \leq b\}$ where $a, b \in U$ and the $\leq$-relation is defined componentwise. The set of all intervals in U is denoted by $\mathcal{I}$. A basic scheme for constraint propagation with

- A constraint set $\mathcal{C} = \{C_1, ..., C_n\}$ over variables $v(1), ..., v(k)$ with domain $U = Dom(v(1)) \times ... \times Dom(v(k))$.
- A selection function $c : \mathbb{N} \to \mathcal{C}$ which selects a constraint C for processing in each step i.
- A propagation function $d : 2^U \times \mathcal{C} \to 2^U$ for constraint propagation which is monotonously decreasing in the first argument: $d(D, C) \subseteq D$.
- A stop function $t : \mathbb{N} \to \{true, false\}$.

works as follows:

Algorithm 1: Basic Scheme for Constraint Propagation, BSCP

```
Input: $D_0 := U, i := 1$
Result: restriction $D$
1 Step(i): $D_i := d(D_{i-1}, c(i))$.
2 If $t(i) = true$: Stop.
3 Otherwise $i := i + 1$, continue with Step(i)
```

The restrictions are used to shrink the search space for possible solutions. If the shrinkage is too strong, possible solutions may be lost. For that, backtracking is allowed in related algorithms.

To keep the scheme simple, the functions c and t depend only on the time step. A basic strategy for c is a round robin over all constraints from $\mathcal{C}$, while more elaborate algorithms use some heuristics. A more sophisticated stop criterion t considers the changes in the sets D_i. Note that the sequence needs not to become stationary if only $D_i = D_{i-1}$. Actually, the sequence $D_0, D_1, D_2, \dots$ needs not to become stationary at all.

For localization problems with simple constraints it is possible to compute the solution directly:

Corollary 6.1. If the propagation function d is defined by $d(D, C) := D \cap C$ for all $D \subseteq U$ and all $C \in \mathcal{C}$, then the sequence becomes stationary after $n = card(\mathcal{C})$ steps with the correct result $D_n = \bigcap \mathcal{C}$.

For simpler calculations, the restrictions D_i are often taken in simpler forms (e.g. as intervals) and the restriction function d is defined accordingly.

Usually constraint satisfaction problems need only some but not necessarily all solutions. For that, the restriction function d need not to regard all possible solutions (i.e. it need not be conservative according to definition 6.2 below). A commonly used condition is local consistency:

Definition 6.1. (Locally consistent propagation function)

1. A restriction D is called **locally consistent w.r.t. a constraint** C if

$$\forall d = [d_1, ..., d_k] \in D \quad \forall i = 1, ..., k \quad \exists d' = [d'_1, ..., d'_k] \in D \cap C \ : d_i = d'_i$$

   i.e. if each value of a variable of an assignment from D can be completed to an assignment in D which satisfies C.

2. A propagation function $d : 2^U \times \mathcal{C} \to 2^U$ is **locally consistent** if it holds for all D, C: $d(D, C)$ is locally consistent for C.

3. The **maximal locally consistent** propagation function $d_{maxlc} : 2^U \times \mathcal{C} \to 2^U$ is defined by $d_{maxlc}(D, C) := Max\{d(D, C) | d \text{ is locally consistent}\}$.

Since the search for solutions is easier in a more restricted the search space (as provided by smaller restrictions D_i), constraint propagation is often performed not with d_{maxlc}, but with more restrictive ones. Backtracking to other restrictions is used if no solution is found.

For localization tasks, the situations is different: We want to have an overview about all possible poses. Furthermore, if a classical constraint problem is inconsistent, then the problem has no solution. In localization problem, there does exist a solution in reality (the real poses of the objects under consideration). The inconsistency is caused e.g. by noisy sensory data. For that, some constraints must be relaxed or enlarged in the case of inconsistencies. This can be done during the propagation process by the choice of even a larger restrictions than given by the maximal locally consistent restriction function.

Definition 6.2. (Conservative propagation function)
A propagation function $d : 2^U \times \mathcal{C} \to 2^U$ is called **conservative** if $D \cap C \subseteq d(D, C)$ for all D and C.

Note that the maximal locally consistent restriction function d_{maxlc} is conservative. We have:

Proposition 6.1. Let the propagation function d be conservative.

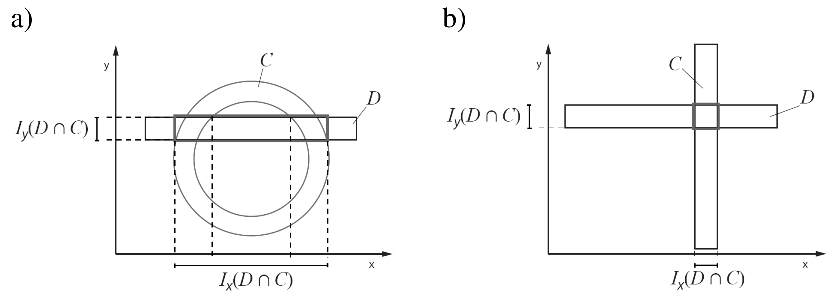

Figure 4. Constraint propagation with intervals D for a) a circular constraint C b) a rectangular constraint C. *Intervals of Projection* w.r.t. $C \cap D$ are illustrated.

1. Then it holds for all restrictions $D_i : \bigcap \mathcal{C} \subseteq D_i$.
2. If any restriction D_i is empty, then there exists no solution, i.e. $\bigcap \mathcal{C} = \emptyset$.

If no solution can be found, then the constraint set is inconsistent. There exist different strategies to deal with that:

- enlargement of some constraints from $\mathcal{C}$,
- usage of only some constraints from $\mathcal{C}$,
- computation of the best fitting hypothesis according to $\mathcal{C}$.

We have discussed such possibilities in Section 5.

As already mentioned above, intervals are often used for the restrictions D, since the computations are much easier. Constraints are intersected with intervals, and the smallest bounding interval can be used as a conservative result. Examples are given in Fig. 4.

Definition 6.3. (Interval Propagation)

1. A propagation function d is called an **interval propagation function** if the values of d are always intervals.
2. The **minimal conservative** interval propagation function $d_{minc} : 2^U \times \mathcal{C} \to \mathcal{I}$ is defined by $d_{minc}(D, C) := Min\{I | I \in \mathcal{I} \wedge D \cap C \subseteq I)\}$ for all D and C.

The results by minimal conservative interval propagation functions can be computed using projections.

Definition 6.4. (Interval of projection)

The (one-dimensional) **Interval of projection w.r.t. to a set** $M \subseteq U$ for a variable v is defined as the smallest interval containing the projection $\Pi_v(M)$ of M to the variable v: $I_v(M) = Min\{I | I \subseteq \mathbb{R} \wedge \Pi_v(M) \subseteq I\}$. It can be computed as $I = [a, b]$ with $a := Min(\Pi_v(M))$ and $b := Max(\Pi_v(M))$.

Both, maximal local consistency and minimal conservatism leads to the same results, and both can be computed using the projections (Figure 4):

Proposition 6.2.

1. $d_{maxlc}(D, C) = d_{minc}(D, C)$

2. $d_{minc}(D, C) = I_{v(1)}(D \cap C) \times .... \times I_{v(k)}(D \cap C)$.

While local consistency is the traditional approach (to find only some solutions), the approach with conservative intervals is more suited for localization tasks because it can be modified w.r.t. to enlarging constraints during propagation for preventing from inconsistency. In case of inconsistencies, the algorithm below is modified accordingly in step 6. The related work is still under investigation.

The following simple and practicable algorithm is used for propagation. The stop condition compares the progress after processing each constraint once. Since stabilization needs not to occur, we provide an additional time limit. Note that the step counting s is not identical to the steps i in the basic scheme BSCP (but could be arranged accordingly).

Algorithm 2: Constraint Propagation with Minimal Conservative Intervals, MCI-algorithm

Input: constraint set $\mathcal{C} = \{C_1, ..., C_n\}$ with variables $\mathcal{V} = \{v_1, ..., v_k\}$ over domain U and a time bound T

Data: $D \leftarrow U$, $s \leftarrow 1$, $D_{old} \leftarrow \emptyset$

Result: minimal conservative k-dimensional interval D

```
1  while $s < T$ & $D \neq D_{old}$ do
2      $D_{old} \leftarrow D$;
3      foreach $C \in \mathcal{C}$ do
4          foreach $v \in \mathcal{V}$ do
5              $D(v) \leftarrow I_v(D \cap C)$;
6          end
7          $D \leftarrow D(v_1) \times \cdots \times D(v_n)$;
8      end
9      $s \leftarrow s + 1$;
10 end
```

Actually, the algorithms works with only single intervals. Looking closer to the possible intersections of constraints (e.g. to the intersection of two circular rings or to the intersection of a circular ring with an rectangle like in Fig. 4a), the sets $D \cap C$ might be better approximated by sets of intervals instead of a single interval. The algorithm could be extended this way: The input and the output for each step are sets of intervals, and all input intervals are processed in parallel. For such purposes the propagation function d of the BSCP could be defined over sets as well. As in other constraint propagation algorithms, it might be lead to better propagation results if we split even a given interval to a union of smaller intervals. But in many cases, when using more constraints, the restrictions will end up with only one of the related intervals anyway.

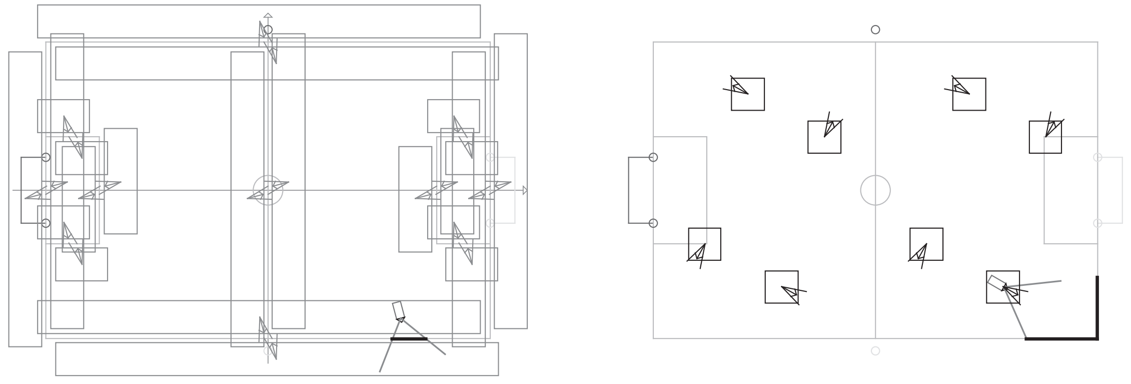

Figure 5. Robot situated on a soccer field. Bold black lines depict the line segment seen by the robot. **(left)** Gray boxes illustrate a constraint generated from the seen line segment. **(right)** Two constraints are generated from perceived lines, black boxes depict the resulting constraint after propagation of these constraints.

6.2. Experimental Results

In our experiments within the RoboCup soccer domain (see section 2), we compared a standard implementation of a Monte-Carlo particle filter (MCPF) with the algorithm described above.

We used constraints given by fixed objects like goalposts, flags and field lines identified in the images by the camera of the robot. It was easy to derive the related constraints: distances to landmarks are defined by circular rings in a generic form, where only the distances derived from the vision system of the robot have to be injected. Constraints given by observed field lines are defined by rectangles and angles, the distances and the horizontal bearings are sufficient to define these constraints. All this can be done automatically. An example for constraints generated from lines and their propagation is given in Fig. 5.

While the particle filter used data from odometry, the constraint approach was tested with the actual vision data only. Since we were able to exploit various redundancies for the MCI, the accuracy of the results were comparable.

Our experiments showed that the MCI algorithm works several times faster than a related particle filter. We performed experiments with a different number of particles. Even with very small sample sets (about 50 particles) the computational costs for the MCPF were several times higher than for MCI. A disadvantage of particle based approaches is that many particles are necessary to approximate the belief which comes at high computational costs.

In further experiments we investigated more ambiguous data (i.e. when only few constraints are available). In this case, the MCI provided a good estimation of all possible positions (all those positions which are consistent with the vision data). The handling of such cases is difficult for MCPF because many particles would be necessary. Related situations may appear for sparse sensor data and for the kidnapped robot problem. Odometry can improve the results in case of sparse data (for MCPF as well as with additional constraints in MCI). But we would argue that the treatment of true ambiguity by MCI is better for the kidnapped robot problem.

7. Conclusion

Constraint propagation techniques are an interesting alternative to probabilistic approaches. From a theoretical point of view, they could help for better understanding of navigation tasks at all. For practical applications they permit the investigation of larger search spaces employing the constraints between various data. Therewith, the many redundancies provided by images and other sensor data can be better exploited.

This paper has shown how sensor data can be transformed into constraints. We presented an algorithm for constraint propagation and discussed some differences to classical constraint solving techniques. In our experiments, the algorithm outperformed classical approaches like particle filters.

We investigated basic concepts including inconsistencies (because data may be noisy) and ambiguity (resulting from under-specified constraint sets). Since the robot is located somewhere in the environment, there is a unique solution in reality – in case of inconsistent constraint sets as well as in case of ambiguous solutions. Our quality measures can be used to handle the trade-off between ambiguity and inconsistency.

Future work will include more investigations on algorithms and further comparisons with existing Bayesian techniques.

References

[1] http://www.ki.informatik.hu-berlin.de, 2007.

[2] http://www.robocup.org, 2007.

[3] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32, 1987.

[4] D. Göhring, K. Gerasymova, and H.-D. Burkhard. Constraint based world modeling for autonomous robots. 2007. Proceedings of the CS&P 2007.

[5] F. Goualard and L. Granvilliers. Controlled propagation in continuous numerical constraint networks. *ACM Symposium on Applied Computing*, 2005.

[6] G. Grisetti, C. Stachniss, S. Grzonka, and Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *RSS*, Atlanta, GA, USA, 2007. Accepted for publiction.

[7] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1998.

[8] M. Jüngel. Memory-based localization. 2007. Proceedings of the CS&P 2007.

[9] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[10] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *International Conference on Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE*, 2006.

[11] A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, editors, *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA-01)*, Lecture Notes in Artificial Intelligence, pages 154–165. Springer, 2001.